DeepRoad: GAN-Based Metamorphic Testing and Input Validation Framework for Autonomous Driving Systems

Mengshi Zhang*
University of Texas at Austin
USA
mengshi.zhang@utexas.edu

Yuqun Zhang†
Shenzhen Key Laboratory of Computational Intelligence, Department of Computer Science and Engineering, Southern University of Science and Technology
China
zhangyq@sustc.edu.cn

Lingming Zhang
University of Texas at Dallas
USA
lingming.zhang@utdallas.edu

Cong Liu
University of Texas at Dallas
USA
cong@utdallas.edu

Sarfraz Khurshid
University of Texas at Austin
USA
khurshid@utexas.edu

ABSTRACT

While Deep Neural Networks (DNNs) have established the fundamentals of image-based autonomous driving systems, they may exhibit erroneous behaviors and cause fatal accidents. To address the safety issues in autonomous driving systems, a recent set of testing techniques have been designed to automatically generate artificial driving scenes to enrich test suite, e.g., generating new input images transformed from the original ones. However, these techniques are insufficient due to two limitations: first, many such synthetic images often lack diversity of driving scenes, and hence compromise the resulting efficacy and reliability. Second, for machine-learning-based systems, a mismatch between training and application domain can dramatically degrade system accuracy, such that it is necessary to validate inputs for improving system robustness.

In this paper, we propose DeepRoad, an unsupervised DNN-based framework for automatically testing the consistency of DNN-based autonomous driving systems and online validation. First, DeepRoad automatically synthesizes large amounts of diverse driving scenes without using image transformation rules (e.g. scale, shear and rotation). In particular, DeepRoad is able to produce driving scenes with various weather conditions (including those with rather extreme conditions) by applying Generative Adversarial Networks (GANs) along with the corresponding real-world weather scenes. Second, DeepRoad utilizes metamorphic testing techniques to check the consistency of such systems using synthetic images. Third, DeepRoad validates input images for DNN-based systems by measuring the distance of the input and training images using their VGGNet features. We implement DeepRoad to test three well-recognized DNN-based autonomous driving systems in Udacity self-driving car challenge. The experimental results demonstrate that DeepRoad can detect thousands of inconsistent behaviors for these systems, and effectively validate input images to potentially enhance the system robustness as well.

CCS CONCEPTS

• **Software and its engineering** → **Software testing and debugging**;

KEYWORDS

Software testing, Test generation, Input validation, Deep Neural Networks

ACM Reference Format:
Mengshi Zhang, Yuqun Zhang, Lingming Zhang, Cong Liu, and Sarfraz Khurshid. 2018. DeepRoad: GAN-Based Metamorphic Testing and Input Validation Framework for Autonomous Driving Systems. In *Proceedings of the 2018 33rd ACM/IEEE International Conference on Automated Software Engineering (ASE '18), September 3–7, 2018, Montpellier, France.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3238147.3238187

1 INTRODUCTION

> The train came out of the long tunnel into the snow country. The earth lay white under the night sky. The train pulled up at a signal stop.
>
> —Yasunari Kawabata, Snow Country

The above quotation is from the first paragraph of fiction "Snow Country", which describes the scene when the protagonist Shimamura enters the snow country. Back to that time, train was the major vehicle for long-distance travels, while people have more choices today. Now, suppose Shimamura takes a Tesla in Autopliot mode [2],

*This work was partially accomplished during visit to Southern University of Science and Technology
†Yuqun Zhang is the corresponding author

ASE '18, September 3–7, 2018, Montpellier, France

ACM ISBN 978-1-4503-5937-5/18/09…$15.00
https://doi.org/10.1145/3238147.3238187

after coming out of the tunnel, there raises a question: can the autopilot system operate safely on the snow-covered road, or the story just ends with a tragedy?

Autonomous driving is expected to transform auto industry. Typically, autonomous driving refers to utilizing sensors (cameras, Radar, Lidar, GPS, etc) [38] to automatically control vehicles without human intervention. The recent advances in Deep Neural Networks (DNNs) enable autonomous driving systems to adapt their driving behaviors according to dynamic environments [2, 14]. In particular, an end-to-end supervised learning framework is made possible to train a DNN for predicting driving behaviors (e.g., steering angles) by inputing driving images, using ⟨driving image, driving behavior⟩ pairs as training data. For instance, DAVE-2 [14], released by NVIDIA in 2016, can accurately predict steering angles based on only images captured by a single front-centered camera of autonomous cars.

Recent testing techniques [31, 38] demonstrate that the autonomous driving systems are error-prone to synthetic images of driving scenes. DeepXplore [31] applies differential testing technique to systematically generate images which disclose the inconsistent behaviors of multiple DNN systems. Specifically, it formulates the image generation problem as a joint optimization problem, which uses gradient-based search techniques to find images for maximizing neuron coverage and the number of inconsistent behaviors of such systems. DeepTest [38] designs systematic ways to automatically generate test cases, seeking to mimic real-world driving scenes. Its main methodology is to transform labeled images of driving scenes by applying simple affine transformations and various effect filters such as blurring/fog/rain to the original images, and check if the autonomous driving systems perform consistently among the original and transformed scenes. With large amounts of original and transformed driving scenes, DeepTest can detect various erroneous driving behaviors for some well-performed open-source autonomous driving models, in a cheap and quick manner.

However, we observe that the methodologies applied in DeepXplore and DeepTest to generate test cases may not accurately reflect the real-world driving scenes, which can rarely contain colored patch or black holes and sidelines; the blurring/fog/rain effects made by simple simulation also appear to be unrealistic, which compromises their efficacy and reliability. For instance, Figure 1 shows the synthetic images which are quoted from the papers of DeepXplore and DeepTest. Note that the colored arrows are attached to present the predicted steering angles. From Figure 1a, 1b and 1c, it can be observed that the images of driving scenes include several artifacts (patch, holes and sidelines), which significantly hurt the image quality. Moreover, for Figure 1d, it appears to be synthesized by simply dimming the original image and mixing it with the scrambled "smoke" effect and it violates the facts that the density of fog varies along depth Similarly, in Figure 1e, DeepTest simply simulates rain by adding a group of lines over the original image. This rain effect transformation is even more distorted because usually when it rains, the camera or front windshield tends to be wet and the image is highly possible to be blurred. These facts show that it is difficult to determine whether the erroneous driving behaviors are caused by the flaws of the DNN-based models, or the inadequacy of the testing technique itself. Furthermore, these transformations (e.g. translation, shear and rotation) can only generate similar images, while they cannot sophisticatedly synthesize images with different styles and thus limit the diversity of test cases. For instance, the snowy road condition demands different complicated transformations for rendering the texture of road and roadside objects (such as trees), and it cannot be generated by simple transformation rules.

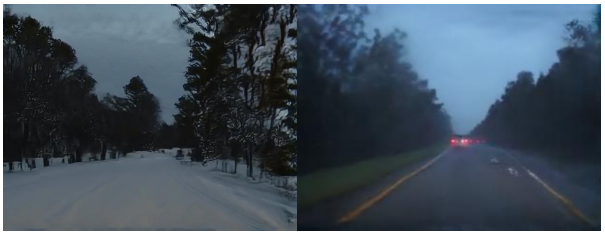
For traditional software, input validation (IV) is an important step before executing programs. For instance, in web applications, IV checks and filters illegal or malicious inputs to prevent application-level attacks such as buffer overflow and code-injection attack [25]. However, to the best of our knowledge, current DNN-based systems lack to validate inputs (e.g., images of driving scenes) and thus tends to cause system vulnerability. Specifically, invalid inputs such as outlier images of driving scenes can highly degrade the prediction accuracy and dramatically increase the risks of DNN-based systems. For example, suppose a DNN-based autonomous driving system is trained on a dataset which only includes images of sunny driving scenes. For out-of-domain inputs (e.g. rainy images of driving scenes) that the system is not trained with, it is highly possible that the system outputs wrong control signals which lead to danger for drivers and passengers.

To address above issues, in this paper, we propose an unsupervised learning framework, namely DeepRoad, to systematically analyze DNN-based autonomous driving systems. DeepRoad is composed of a metamorphic testing module, DeepRoad$_{MT}$ and an input validation module, DeepRoad$_{IV}$. DeepRoad$_{MT}$ employs a Generative Adversarial Network (GAN)-based technique [18, 27] to synthesize driving scenes with various weather conditions, and develops a metamorphic testing module for DNN-based autonomous driving systems. Specifically, the metamorphic relations are defined such that no matter how the driving scenes are synthesized to cope with different weather conditions, the driving behaviors are expected to be consistent with those under the corresponding original driving scenes. At this point, DeepRoad$_{MT}$ enables us to test the accuracy and reliability of DNN-based autonomous driving systems under different scenarios, including heavy snow and hard rain, which can greatly complement the existing approaches (e.g., DeepXplore, DeepTest). For instance, Figure 2 presents the snowy and rainy scenes generated by DeepRoad$_{MT}$ (from sunny scenes), which can hardly be distinguished from genuine ones and cannot be generated using simple transformation rules. DeepRoad$_{IV}$ is designed to validate inputs for DNN-based autonomous driving systems based on image similarity. Firstly, DeepRoad$_{IV}$ applies a pre-trained DNN model–VGGNet to extract high-level features (i.e. contents and styles) of both training and test input images. Then, the Principle Component Analysis (PCA) technique is applied on these features for dimension reduction. Finally, DeepRoad$_{IV}$ validates inputs by comparing the average distance between training and input images with a preset threshold.

To evaluate the effectiveness of DeepRoad, we first synthesize driving scenes under heavy snow and hard rain. In particular, based on GAN, we collect images with two extreme weather conditions from YouTube videos to transform real-world driving scenes, and deliver them with the corresponding weather conditions. Subsequently, these synthetic scenes are used to test three open-source DNN-based autonomous driving systems from Udacity community [7]. The experimental results reveal that DeepRoad$_{MT}$ can effectively detect thousands of inconsistent behaviors of different levels for these systems. Furthermore, we use DeepRoad$_{IV}$ to validate the input images

(a) Patch (b) Holes (c) Translation (d) Fog (e) Rain

Figure 1: Driving scenes synthesized by DeepXplore (a)(b) and DeepTest (c)(d)(e)



(a) Snow (b) Rain

Figure 2: Snowy and rainy scenes synthesized by DeepRoad

sampled from different driving scenes. The results demonstrate that in the embedding space, the cluster of the rainy and snowy image points are separately distributed to the cluster of training images, however, the training cluster is mixed with the majority of the sunny image points. It indicates that given a proper threshold, DeepRoad$_{IV}$ can effectively validate input, which potentially improve the system robustness.

The key contributions of this paper are as follows.

- We propose the first GAN-based metamorphic testing approach to generate driving scenes with various weather conditions for detecting inconsistent behaviors of autonomous driving systems.
- We propose a novel approach to validate inputs for DNN-based autonomous driving system. We present that the distance between the high-level features of training and input images can be used for validating inputs.
- We implement the proposed approaches in DeepRoad, which can generate images of diverse driving scenes (e.g. rain and snow) and measure the similarity between multiple image sets in embedding space. We use DeepRoad to test well-recognized DNN-based autonomous driving models and successfully detect thousands of inconsistent driving behaviors. Additionally, DeepRoad can accurately distinguish images with extreme weather conditions to the training images, which is effective to validate input for autonomous driving systems.

2 BACKGROUND

2.1 Deep Neural Networks for Autonomous Driving

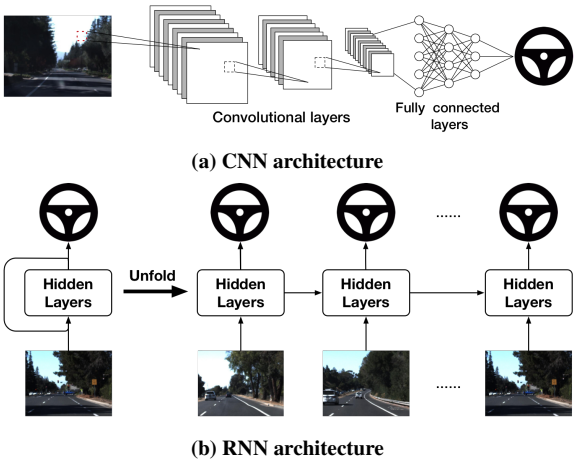
Autonomous driving systems have been rapidly evolving in recent years [14, 32]. For example, many major auto manufacturers (including Tesla, GM, Volvo, Ford, BMW, Honda, and Daimler) and IT companies (including Waymo/Google, Uber, and Baidu) are working on building and testing various autonomous driving systems. Typically, autonomous driving systems capture data from environment via multiple sensors (e.g. camera, Radar, Lidar, GPU, IMU, etc.) as input, and use Deep Neural Networks (DNNs) to process data and output control signals (e.g. steering and braking decisions). In NVIDIA's work [14], their autonomous driving system, DAVE-2 can fluently control cars only based on the images captured by a single front camera. In this work, we mainly focus on DNN-based autonomous driving systems with camera inputs and steering angle outputs.

2.2 DNN Architectures

To date, Convolutional Neural Network (CNN) [23] and Recurrent Neural Network (RNN) [33] are the most widely used DNNs for autonomous driving systems. Typically, CNNs are good at analyzing visual imagery and RNNs can effectively process sequential data. In this work, the evaluated models are built on CNN and RNN modules. We briefly introduce the basic concepts and components of each architecture as follows, where more details about DNNs are provided in [24].

2.2.1 Convolutional Neural Networks. Convolutional Neural Networks are similar to regular neural networks, which include a large amount of neurons and pass information in a feed-forward way. However, since the input data are images, several properties can be applied to optimize the regular neural networks, where convolutional layer is a key component in CNNs. Instead of being fully connected, a neuron in a layer only connects to some neurons in the previous layer, and the computational process can be presented as a convolution with kernels. Figure 3a shows an example of CNN-based autonomous driving system that consists of an input layer (images) and an output layer (steering angles), as well as multiple hidden layers. Convolution hidden layers allow weight sharing across multiple connections and can greatly save the training efforts.

2.2.2 Recurrent Neural Networks. Regular neural networks and CNNs are designed to process independent data, such as using CNN to classify images. However, for sequential data like videos, the neural networks should not only capture information of each single frame, but are also expected to model the connections between them. Unlike regular NNs and CNNs, RNN is a kind of neural network with feedback connections. As shown in the left part of Figure 3b, RNNs use loops to forward the previous states to input, which model the connection of input data. The right part of Figure 3b shows the workflow of the unfolded RNN for predicting steering angles based on a sequence of images. At each step, RNN takes the current input



(a) CNN architecture

(b) RNN architecture

Figure 3: Autonomous driving systems built on CNN and RNN

image and previous hidden states as input and predicts the steering angle.

2.3 Challenge of Testing for DNN-based Autonomous Driving Systems

DNN-based autonomous driving systems are essentially software systems, which are error-prone and can lead to tragedies. For example, on January, 2018, a Tesla Model S plowed into a fire truck at 65 mph while using Autopilot [9]. And on Mar, 2018, an autonomous Uber failed to slow down and killed a pedestrian during road test at night [10]. To ensure the quality of software systems, many software testing techniques have been proposed in the literature [11, 29], where typically, a set of specific test cases are generated to test if the software programs perform as expected. The process of determining whether the software program performs as expected upon the given test inputs is known as the *test oracle* problem [11]. Despite the abundance of traditional software testing techniques, they cannot be directly applied for DNN-based systems since the logics of DNN-based softwares are learned from data with minimal human interference (like a blackbox) while the logics of traditional software programs are manually created.

3 APPROACH

3.1 Metamorphic Testing for DNN-based Autonomous Driving Systems

3.1.1 Metamorphic DNN Testing. Metamorphic Testing [35] (MT) has been widely used to automatically generate tests to detect software faults. The strength of MT lies in its capability to automatically solve the test oracle problem via Metamorphic Relations (MRs). In particular, let p be a program mathematical representation that maps program inputs to program outputs (e.g., $p[\![i]\!] = o$). Assuming f_I and f_O are two specific functions for transforming the input and output domain respectively, and they satisfy the following MR formulation:

$$\forall i,\ p[\![f_I(i)]\!] = f_O(p[\![i]\!]) \tag{1}$$

, where i denotes the input of program p.

With such MRs, we can test a specific implementation $\hat{p}$ of p by checking whether $\hat{p}[\![f_I(i)]\!] = f_O(\hat{p}[\![i]\!])$ for various input i. Accordingly, MT is defined as testing a program implementation via cross-checking inputs and outputs with MRs. For instance, given a program implementing function *sine*, MT can be used to delineate test oracles and create various new tests. For any existing input i to test function *sine*, various facts can serve as MRs, e.g., $\sin(-i) = -\sin(i)$ and $\sin(i + 2\pi) = \sin(i)$. These facts can be formulated as 1) $f_I(x) = f_O(x) = -x$, 2) $f_I(x) = x + 2\pi$ and $f_O(x) = x$. With such MRs, we can transform the existing test inputs according to f_I to generate additional tests, and check the output based on f_O. For instance, suppose the default test case of function *sine* is `AssertTrue(sin(0.5·Pi), 1.0)`. Based on above MRs, we can generate two extra tests: `AssertTrue(sin(-0.5·Pi), -1.0)` and `AssertTrue(sin(2.5·Pi), 1.0)`.

In this work, we further apply MT to test DNN-based autonomous driving systems. Formally, denote DNN as a DNN-based autonomous driving system that continuously maps each image into predicted steering angle signal (e.g., turning left for 15°). One MR can be defined as given the original image stream $\mathbb{I}$, various image transformations $\mathbb{T}$ can simply change the road scenes (detailed shown in Section 3.1.2) without impacting the predictions for each image $i \in \mathbb{I}$ (e.g., the predicted direction should be approximately the same on the same road under different weather conditions). This MR to test DNN with additional transformed inputs can be formalized as follows:

$$\forall i \in \mathbb{I} \wedge \forall \tau \in \mathbb{T},\ DNN[\![\tau(i)]\!] = DNN[\![i]\!] \tag{2}$$

3.1.2 DNN-based Road Scene Transformation. The recent work DeepTest [38] also applied MT to test DNN-based autonomous driving systems. However, it only performs basic image transformations, such as adding simple blurring/fog/rain effect filters, and thus has the following limitations: (1) DeepTest may generate images which violate common scenes (discussed in Section 1). (2) DeepTest cannot simulate complex road scene transformations (e.g., snowy scenes).

To complement DeepTest by automatically generating various real-world road scenes, in this work, we leverage UNIT [27], a recent published DNN-based method to perform unsupervised image-to-image transformation based on Generative Adversarial Networks (GANs) [18] and Variational Autoencoders (VAEs) [22]. One insight of UNIT is that suppose two images contain the same contents but lie in different domains, they should have the same representations in a shared-latent space. Accordingly, given a new image from one domain (e.g., the original driving scene), UNIT can automatically generate its corresponding version in the other domain (e.g., rainy driving scene).

Figure 4 [27] presents the structure of UNIT, S_1 and S_2 denote two different domains (e.g., sunny and rainy driving scenes), E_1 and E_2 denote two autoencoders which project the images from S_1 and S_2 to a shared-latent space Z. Suppose x_1 and x_2 are paired images which share the same content. Ideally, E_1 and E_2 would encode them to the same latent vector z, and it can be translated back to S_1 and S_2 by two domain specific generators G_1 and G_2, respectively. D_1 and D_2 are two discriminators which detect whether the image belongs to S_1 and S_2 respectively. Specifically, they are expected to differentiate whether the input image is sampled from target

domain (e.g. real image) or produced by a well-trained generator (e.g. synthetic image). Based on the autoencoders and generators, UNIT can be used to transform images between two domains. For instance, image x_1 can be transformed to S_2 by $G_2(E_1(x_1))$.

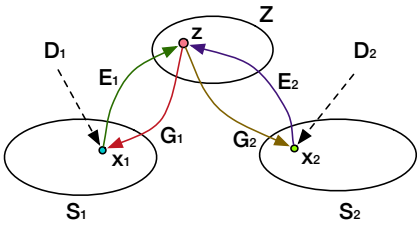


Figure 4: Structure of UNIT

In UNIT, all D_i, E_i and G_i are incarnated as neural networks, and the learning objective of UNIT can be decomposed to optimize the following costs:

- **VAE loss:** minimizing the loss of image reconstruction for each $\langle E_i, G_i \rangle$.
- **GAN loss:** achieving the equilibrium point in the minimax game for each $\langle G_i, D_i \rangle$, where D_i aims at discriminating the images to find out whether they are sampled from the domain S_i or produced by G_i that aims at fooling D_i.
- **Cycle-consistency loss:** minimizing the loss of cycle- reconstruction for each $\langle E_i, G_j, E_j, G_i \rangle$, where x_1 is expected to equal to $G_1(E_2(G_2(E_1(x_1))))$ and x_2 is expected to equal to $G_2(E_1(G_1(E_2(x_2))))$.

The total loss can be summarized as follows:

$$\begin{aligned}\min_{E_1,E_2,G_1,G_2}\max_{D_1,D_2} \mathcal{L}_{CC_1}(E_1,G_2,E_2,G_1)\\ +\mathcal{L}_{CC_2}(E_2,G_1,E_1,G_2)\\ +\mathcal{L}_{VAE_1}(E_1,G_1)+\mathcal{L}_{VAE_2}(E_2,G_2)\\ +\mathcal{L}_{GAN_1}(D_1,G_1)+\mathcal{L}_{GAN_2}(D_2,G_2)\end{aligned}$$

, and this loss function can be optimized using Stochastic Gradient Descent algorithm.

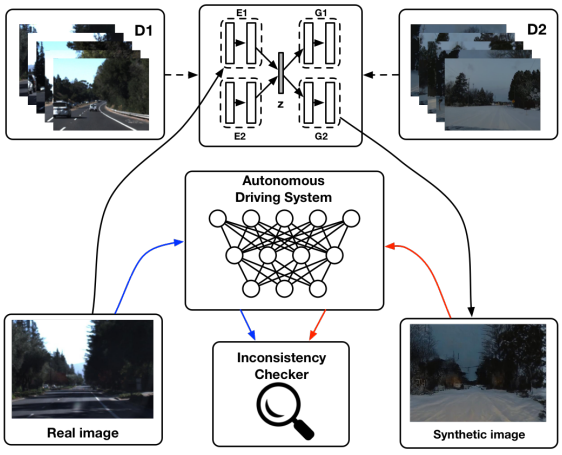


Figure 5: Framework of DeepRoad$_{MT}$

3.1.3 Framework of DeepRoad$_{MT}$. Figure 5 shows the overall design of our metamorphic testing framework for DNN-based autonomous driving systems–DeepRoad$_{MT}$. In Figure 5, DeepRoad$_{MT}$ first takes *unpaired* training images from two target domains (e.g., datasets of the driving scene under sunny and snowy weather respectively), and utilizes UNIT to project two domains to the same latent space by optimizing the loss functions presented in Section 3.1.2. When the training process finished, DeepRoad$_{MT}$ uses the well-trained model to transform the whole dataset of sunny driving scenes to snowy weather. Specifically, given any image under sunny weather i, DeepRoad$_{MT}$ encodes it to vector z_i by E_1, and synthesizes its corresponding version under snowy weather $\tau(i)$ using G_2. DeepRoad$_{MT}$ feeds each pair of real and synthetic driving scene images to the autonomous driving systems under test, i.e., DNN, and compare their prediction results $DNN[\![\tau(i)]\!]$ and $DNN[\![i]\!]$ to detect any inconsistent behaviors. Normally, the transformed driving scenes are expected not to significantly impact the predicted steering angles, and any inconsistency may indicate correctness or robustness issues of the systems under test [31, 38].

3.2 Input Validation for DNN-based Autonomous Driving Systems

Driving scenes synthesized by DeepTest and DeepXplore can be used as test cases to test DNN-based autonomous driving systems in an offline manner. Though these test cases are useful to expose the system vulnerability and advise developers to complement training data from real world to improve the system robustness, it is not sufficient for online testing. For instance, a DNN-based autonomous driving system can be well trained and perfectly function in sunny environments, yet it might perform incorrectly at night or on a snow-covered road, because the lane marks it detected for guiding cars disappear in such driving scenes. This example suggests that if the system can validate input images online, and actively advise drivers to control the car when it cannot handle the invalid inputs, the autonomous driving systems can become safer and more robust. In the following, we first define the criteria of input validation for DNNs (especially image-oriented models), and present our input validation framework for DNN-based autonomous driving systems.

3.2.1 Input validation of DNNs. The goal of input validation (IV) is to ensure that only properly formed data can be accepted by systems, and malformed data should be rejected before execution. The reason is that an invalid input may trigger malfunction of downstream components, which makes the system insecure. Generally, the valid input of a program can be explicitly defined such as the input string should not be null/empty or the value of a certain input variable should be greater than 0. However, it is not trivial to properly define input validity of a DNN-based program. For example, we can define an IV criteria as the input data should be any RGB image with size `640*480`, or any input data should exist in the training dataset to guarantee the correctness. However, none of them are proper since the first criteria is too weak to improve system robustness, and the second is so strong that makes the system lack generalisability.

We define the IV criteria of DNN-based program based on the Probably Approximately Correct (PAC) Learning theory. According to the PAC Learning theory [13], a machine learning model Λ is expected to learn the distribution $\mathscr{D}$ from the training dataset,

and predict the correct label with high probability. [1] This can be formulated as follows:

$$E(\Lambda; \mathscr{D}) = Pr_{\mathbf{x}\sim\mathscr{D}}(\Lambda(\mathbf{x}) \neq y) \tag{3}$$

$$Pr(E \leq \epsilon) \geq 1 - \sigma \tag{4}$$

In Formula 3, E denotes that the probability of Λ makes incorrect prediction ($\Lambda(\mathbf{x}) \neq y$) on input data $\mathbf{x}$ sampled from $\mathscr{D}$, and in Formula 4, ϵ and σ are two parameters between 0 and 1, such that it is highly possible (greater that $1-\sigma$) E is small (less than ϵ), which means Λ is effective on $\mathscr{D}$. Based on the above equations, we first define an abstract IV criteria of DNN-based systems is that the input data should be sampled from $\mathscr{D}$. As discussed before, suppose the input data is not sampled from $\mathscr{D}$, the IV criteria is violated and the prediction accuracy is not guaranteed. Therefore, it is necessary to validate inputs to improve the robustness of DNN-based systems.

Intuitively, the IV criteria should be instantiated as:

$$Pr_{\mathbf{x}\sim\mathscr{D}}(\mathbf{x} = i) > \theta \tag{5}$$

, which means the probability of input i being sampled from $\mathscr{D}$ should be greater than the predefined threshold θ. Otherwise, the system refuses to predict on i. However, this definition is not tractable for image data, because image data is highly dimensional and their distribution (e.g. Gaussian Mixture model) is difficult to be explicitly represented. To address this issue, we project image data to a low-dimensional space and use the distance between inputs and training data to replace $Pr_{\mathbf{x}\sim\mathscr{D}}(\mathbf{x} = i)$. In particular, according to the Manifold Learning theory [13], the images generated by $\mathscr{D}$ can be embedded into a non-linear low-dimension manifold $M_{\mathscr{D}}$. Suppose the input data i is sampled from $\mathscr{D}$, its projection i_p should be included by $M_{\mathscr{D}}$. Furthermore, we propose an extra constraint for the non-linear embedding that suppose the input data are generated by a different distribution $\mathscr{D}'$, their projections are expected to be included by another manifold $M_{\mathscr{D}'}$, which is linearly separable to $M_{\mathscr{D}}$. Based on the constraint, we can compute the minimal distance of i_p and the projections of training data to validate if i_p belongs to $M_{\mathscr{D}}$. The IV criteria is redefined as follows:

$$\min_j \|h(i) - h(j)\|_2 < \theta' \tag{6}$$

, where $\|\cdot\|_2$ denotes L_2 norm, $h(\cdot)$ denotes the required non-liner projection and θ' denotes predefined threshold for input validation. If the input satisfies Equation 6, it will be processed by DNNs for prediction, otherwise, it will be rejected.

3.2.2 *Framework of DeepRoad$_{IV}$.*

We propose DeepRoad$_{IV}$, an input validation framework for autonomous driving systems. DeepRoad$_{IV}$ separate the projection $h(\cdot)$ to two parts: non-linear transformation and dimension reduction. For the first part, DeepRoad$_{IV}$ applies VGGNet [37], a widely used DNN [17, 20] to extract high-level features from each image. To be specific, the input image is encoded in each layer of VGGNet by kernels. Suppose layer i includes N_i distinct kernels, it generates N_i feature maps each of size (w_i, h_i), where w_i and h_i are the width and height of the feature maps respectively. These feature maps can be stored as a feature matrix F^i with size (N_i, M_i), where each row of F_i is the vector flattened from the corresponding feature map and M_i is $w_i \cdot h_i$.

[1] For simplicity, here we only discuss DNNs for classification, the same approach can be applied to explain DNNs for other tasks such as regression.

DeepRoad$_{IV}$ also generates style information which are introduced in [16]. These style information aims at capturing the texture of images and it is defined by feature correlation, which can be computed by the Gram matrix

$$G_i = F_i \cdot F_i^T \tag{7}$$

Suppose we choose layer i and j to extract the feature and style matrix F_i and G_j respectively, the representation vector of the given image is $\vec{v} = [\vec{v_F}, \vec{v_G}]$, where $\vec{v_F}$, $\vec{v_G}$ are flattened vector of F_i and G_j receptively. Further, we apply Principle Component Analysis (PCA) technique to reduce feature dimension of input and training data as follows:

$$Y = X \cdot P \tag{8}$$

X denotes the input matrix with size (n, m), where n is the total number of input and training data and m is the length of feature vector $\vec{v}$. P denotes the projection matrix with size (m, k), where k is the target dimension less than m, and P can be computed using X [13].

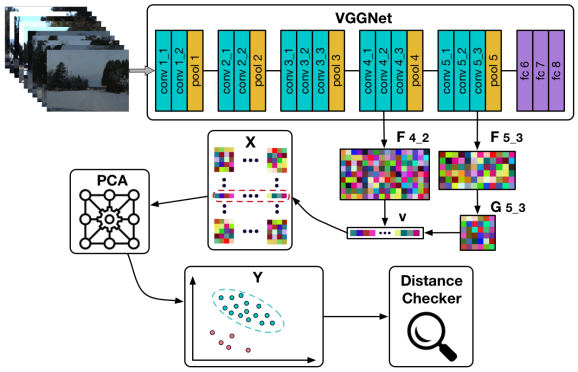


Figure 6: Framework of DeepRoad$_{IV}$

Figure 6 shows the overall design of our input validation framework, DeepRoad$_{IV}$. DeepRoad$_{IV}$ first takes the training and online driving images as input, and uses VGGNet to extract their content and style features. As shown in Figure 6, DeepRoad$_{IV}$ inputs a snowy image to VGGNet, and chooses the convolutional layer `conv 4_2` and `conv 5_3` to extract content and style features respectively. To be specific, the colored grids `F 4_2` and `F 5_3` denote the content features extracted from VGGNet, and the style feature `G 5_3` is computed by Equation 7. Note that these colored grids are just used to visualize results, and their dimensions do not match the real outputs. Then, matrix `F 4_2` and `G 5_3` are flattened and concatenated to feature vector $\mathbf{v}$. DeepRoad$_{IV}$ processes all image data using the same approach and the feature vectors compose matrix $\mathbf{X}$. In the second step, DeepRoad$_{IV}$ applies PCA to reduce the feature dimension. In Figure 6, we set the target dimension to 2. The processed data Y are presented on a 2-D plane, where the blue and red nodes denote the training and online driving images respectively. Finally, DeepRoad$_{IV}$ computes the minimal distance between training data and each online image, and refuses to predict for the images whose distances are greater than a certain threshold.

Table 1: Details of image sets

Dataset	Frame	Duration	Weather Cond.
Udacity Training	33805	N.A.	Sunshine
Udacity Test Ep1	15212	N.A.	Sunshine
Udacity Test Ep2	5614	N.A.	Sunshine
Youtube Ep1	1000	28:55	Heavy snow
Youtube Ep2	1000	1:09:03	Hard rain

4 EXPERIMENTS

4.1 Data

We use a real-world dataset released by Udacity [8] as a baseline to check the inconsistency of autonomous driving systems. From the dataset, we select two episodes of high-way driving video where obvious changes of lighting and road conditions can be observed among frames. To train UNIT model, we also collect images of extreme scenarios from YouTube. In the experiments, we select heavy snow and hard rain, two extreme weather conditions to transform real-world driving scenes. To make the variance of collected images relatively large, we only search for videos which is longer than 20 mins. In the scenario of hard rain, the video records wipers swiping windshield, which would potentially degrade the quality of synthetic images. Hence, in data preprocessing phase, we manually check and filter those images. Note that all images used in the experiments are cropped and resized to `320*240`, and we have performed downsampling for YouTube videos to skip consecutive frames with close contents. The detailed information is present in Table 1.

4.2 Models

We evaluate our metamorphic testing framework DeepRoad$_{MT}$ on three DNN-based autonomous driving models which are released by Udacity [8]: `Autumn` [4], `Chauffeur` [5], and `Rwightman` [6]. We choose these three models as their pre-trained models are public and can be evaluated directly on the synthetic datasets. To be specific, the model details of `Rwightman` are not publicly available, however, similar to black-box testing, our approach aims at detecting the inconsistencies of the model. Hence, `Rwightman` is still used for evaluations.

Autumn. `Autumn` is composed of a data preprocessing module and a CNN. Specifically, `Autumn` first computes the optical flow of raw images and inputs them to a CNN to predict steering angles. The architecture of `Autumn` is: three 5x5 conv layers with stride 2 pluses two 3x3 conv layers and followed by five fully-connected layers with dropout. The model is implemented by OpenCV, Tensorflow and Keras.

Chauffeur. `Chauffeur` consists of one CNN and one RNN with LSTM module. The workflow is that CNN first extracts the features of input images and then utilizes RNN to predict the steering angle from previous 100 consecutive images. This model is also implemented by Tensorflow and Keras.

4.3 Metric

Metric of model inconsistency. In this work, an autonomous driving system is defined to act consistent if its steering angle prediction falls within certain error bounds after modifying the weather condition of driving images. We define the number of inconsistent behaviors of autonomous driving systems as follows:

$$IB(DNN, \mathbb{I}) = \sum_{i \in \mathbb{I}} f(|DNN[\![i]\!] - DNN[\![\tau(i)]\!]) > \epsilon)$$

,where DNN denotes the autonomous driving model and $\mathbb{I}$ is the real-world driving dataset. i denotes the ith image in $\mathbb{I}$. τ denotes the image generator/transformer which can change the weather condition of the input image. f is an indicator function which outputs 1 or 0 if and only if the input is $True$ or $False$ and ϵ is the error bound.

Metric of input validation. As introduced in Section 3.2.2, the input validity of DNN-based autonomous driving systems is defined by the minimal distance of input and training images in the embedding space. This metric can reflect the similarity between the input and training data, however, it has the following limitations: first, generally, the training dataset is large (e.g. 10k images). Suppose we use the above metric to validate a single input image, the numerous training data points will dominate PCA and the results are biased. Second, using the minimal distance for input validation is not stable. For example, suppose the distance of input i and training data j is minimal and it is less than the threshold. However, j is far from other training data and actually i is not similar to the majority of the training dataset. We address these limitations in the following ways: first, to balance the input data and training data, we collect M images from online driving scenes as input data, and randomly select M images from training dataset as training data. Second, to estimate the distance more stable, we average the Top-N minimal distances of each image to represent their similarities. The metric of input validity is defined as follow:

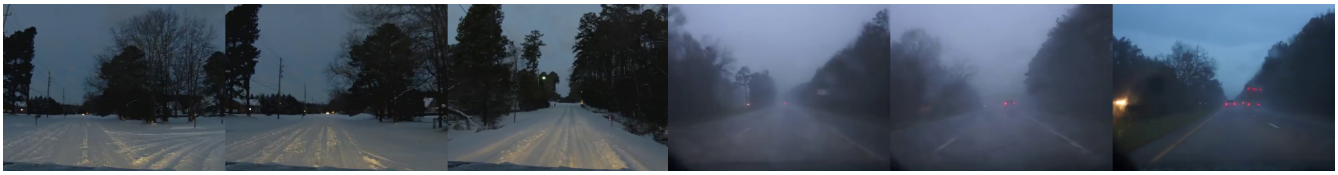
$$m_{IV}(i, S_t) = f(\frac{1}{N} \sum_{k \in \{N\}} \min^k_{j \in S_t} (\|h(i) - h(j)\|_2) < \theta)$$

, where N is a parameter less than M, i denotes the image of the input dataset with size M. S_t denotes the set of M randomly selected training images, $min^k(\cdot)$ denotes the k-th minimal value among input array. Function f is an indicator function and θ is the threshold of input validation.

4.4 Results

4.4.1 Results of DeepRoad$_{MT}$. We first present several YouTube screenshots as ground truth in Figure 7 to help readers check the quality of synthetic images. In Figure 8, we list real and GAN-generated images pairs, where the two rows present the transformation of Udacity dataset to snowy and rainy scenes, respectively, and the odd and even columns present original and GAN-generated images, respectively. Qualitatively, the GAN-generated images are visually similar to the images collected from YouTube videos and they also can keep the major semantic information (such as the shape of tree and road) of the original images. Interestingly, in the first snowy image in Figure 8, the sky is relatively dark and GAN can successfully render the snow texture and the light in front of the car. In the second column, the sharpness of rainy images are relatively low and this is consistent to the real scene showed in Figure 7. Our results are consistent with the original UNIT work [27], and further demonstrate the effectiveness of UNIT for image transformation.

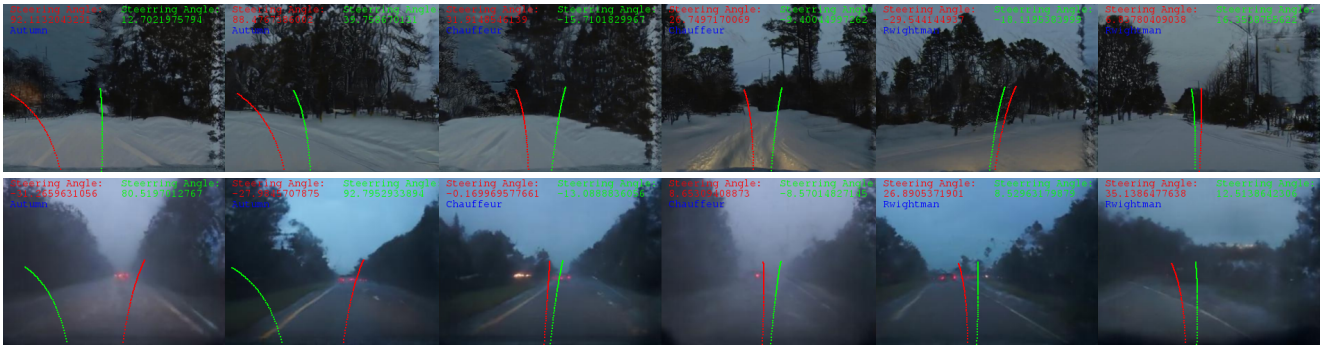
We further present examples for the detected inconsistent behaviors of autonomous driving models in Figure 9. In the figure,



(a) Heavy snow (b) Hard rain

Figure 7: Images collected from YouTube

Figure 8: Real and GAN-generated images.

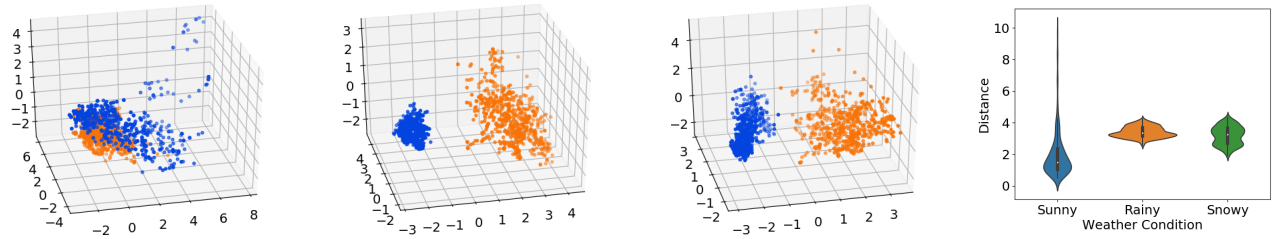


(a) Autumn (b) Chauffeur (c) Rwightman

Figure 9: Inconsistency of steering angle prediction on real and synthetic images.

each row shows the scenes of snow and rain, respectively. In each sub-figure, the blue caption indicates the model name, while the red and green captions indicate the predicted steering angles on the real and synthetic images, respectively. The curves visualize the predictions which help check the differences. From the figure we can observe that model `Autumn` (the first two columns) has the highest inconsistency number on both scenes; in contrast, model `Rwightman` (the last two columns) is the most stable model under different scenes. This figure shows that DeepRoad$_{MT}$ is able to find inconsistent behaviors under different road scenes for real-world autonomous driving models. For example, a model like `Autumn` or `Chauffeur` [3] (they are both ranked higher than `Rwightman` in the Udacity challenge) may work perfectly in a sunny day but can crash into the curbside (or even worse, the oncoming cars) in a rainy or snowy day (shown in Figure 9).

Table 2 presents the detailed number of detected inconsistent behaviors under different weather conditions and error bounds for each studied autonomous driving model on the Udacity dataset. For example, when using the error bound of 10° and the rainy scenes, DeepRoad$_{MT}$ detects 5279, 710, and 656 inconsistent behaviors for `Autumn`, `Chauffeur`, and `Rwightman`, respectively. From the table we can observe that the inconsistency number of `Autumn` is the highest under both weather conditions. We think one potential reason is that `Autumn` is purely based on CNN, and does not utilize history information (e.g., via RNN), and thus may not always perform well in all road scenes. On the other hand, `Rwightman` performs the most consistently than the other two models under all error bounds. This result presents a very interesting phenomenon – DeepRoad$_{MT}$ can not only detect thousands of inconsistent behaviors of the studied autonomous driving systems, but can also measure different autonomous systems in terms of their robustness. For example, with the original Udacity dataset, it is hard to find the limitations of autonomous driving systems like `Autumn`.



(a) Sunny (b) Rainy (c) Snowy (d) Distribution of Distances

Figure 10: Results of DeepRoad$_{IV}$: Image embeddings and Distance distributions.

Table 2: Number of inconsistency behavior of three models under different weather conditions

Scene	Model	Num. of Inconsist. Behav.			
		10°	20°	30°	40°
Snowy	Autumn	11635	11602	11388	10239
	Chauffeur	4839	2105	1093	653
	Rwightman	334	115	45	14
Rainy	Autumn	5279	5279	5279	5279
	Chauffeur	710	175	94	71
	Rwightman	656	92	23	0

4.4.2 Results of DeepRoad$_{IV}$. We use sunny, rainy and snowy driving scenes to test DeepRoad$_{IV}$. The expectation of this experiment is, in the embedding space, the sunny images are close to the training images, and the rainy and snowy images are linearly separable to them. Specifically, sunny images are collected from the original test dataset, and the rainy and snowy images are extracted from YouTube videos. Note that to ensure the authenticity of input images, we only choose real-world instead of synthetic images. Moreover, we choose convolutional layer `conv 3_2` and `conv 4_1` of VGGNet to extract the content and style features from the input images, and we set the PCA dimension to 3 for visualizing experimental results. To reduce the computational complexity, we resize all the images to `120*90` and set the sampling number M of each dataset to 600. Furthermore, we use the average of Top-100 minimal distances of each data point to reduce the variance of similarity estimation for each input image.

Figure 10 visualizes the results of DeepRoad$_{IV}$ on sunny, rainy and snowy driving scenes. To be specific, the first three figures of Figure 10 present the results of sunny, rainy and snowy images, respectively. And the orange and blue points present the sampled training and corresponding input images. We first analyze the results of the image embedding. From Figure 10a, we observe that the majority of the input images are mixed with the training samples, and a few inputs are far from the cluster. From Figure 10b and 10c, there are gaps between the input and training points and the clusters are linearly separable. These results indicate that the distributions of sunny and training images are close but the rainy and snowy images are not. On the other hand, the cluster of rainy and snowy images are relatively compact but the sunny images are scattered. The reason may be the texture of rainy and snowy images are unified and the content is relatively poor, so that the distances between images are small. However, the light condition and content of sunny images are more diverse, hence the distances are large. Moreover, from Figure 10d, we find the distances of sunny images mainly lie between 0 and 3, and almost all of the distances of rainy and snowy images are larger than 2. Suppose the threshold of input validation is 2.5, DeepRoad$_{IV}$ can detect 100% of rainy, 85% of snowy images and 21% outliers among sunny images as invalid inputs, which effectively improve the system robustness. Furthermore, we study if the non-linear transformation of input images is necessary for input validation. Figure 11 visualizes the results of DeepRoad$_{IV}$ without feature extraction. From Figure 11, we observe that all blue clusters are surrounded by the orange points, which show that input images are not linearly separable to the training images in the embedding space. It implies in this case, the distance is not a proper metric for input validation, and non-linear transformation (i.e. feature extraction using VGGNet) is indeed needed.

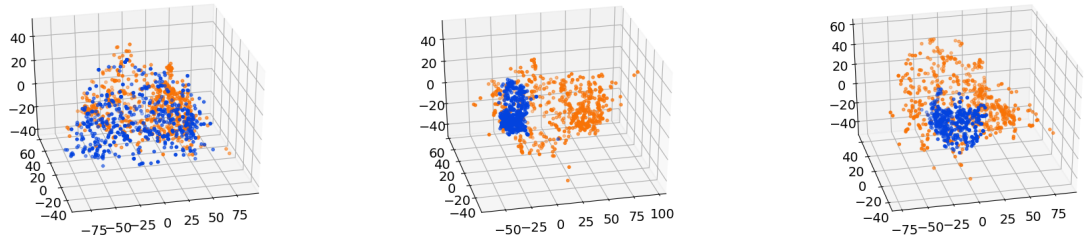
5 THREATS TO VALIDITY

There are several threats to the validity of the proposed approach and its result, which include the followings.

In this work, the main threat to internal validity is potential defects in the implementation of our techniques. To reduce these threats, in implementing DeepRoad$_{MT}$, we used the original implementation of UNIT to ensure DeepRoad$_{MT}$'s performance. Furthermore, in implementing of DeepRoad$_{IV}$, we downloaded the pre-trained VGGNet weights from PyTorch website[2] instead of training it on ImageNet.

The threats to external validity mainly lie in image quality, dataset and autonomous driving models. First, we lack a good standard to evaluate image quality (i.e. realisticity). In this paper, we present GAN-generated images to let readers check their quality. This approach is quite straightforward but less objective. Salimans et.al [34] proposed Inception Score to evaluate the quality of synthetic images. To be specific, Inception Score uses an Inception-v3 Network pre-trained on ImageNet to compute a statistic of the network's outputs as the quality of generated images. However, Barratt et.al [12] demonstrate that Inception Score fails to provide useful guidance when comparing generative models (e.g. GANs). Furthermore, the generation process of GANs is not controllable that some semantic content (e.g. trees or cars) may be missing in synthetic images, and this may threaten the validity of Metamorphic Testing. Second, the Udacity dataset is relative small and the autonomous driving models are quite simple. Suppose the dataset is sufficiently large, a more complicated and robust model is able to be trained, and the inconsistent behaviors would be dramatically reduced. Moreover, an

[2]https://download.pytorch.org/models/vgg19-dcbb9e9d.pth



(a) Sunny (b) Rainy (c) Snowy

Figure 11: Results of DeepRoad$_{IV}$ (without non-linear transformation): Image embeddings and Distance distributions.

autonomous driving system is complicated, and its input and output are diverse. In this work, we only focus on testing the accuracy of the steering angle instead of speeding adjustments.

6 RELATED WORK

Metamorphic testing. Metamorphic testing is a classical software testing method that identifies software bugs [15, 36, 44]. Its key idea is to detect violations of domain-specific metamorphic relations defined across outputs from multiple runs of the program with different inputs. Metamorphic testing has been applied for testing machine learning classifiers [30, 40, 41]. In this paper, DeepRoad develops a specific GAN-based metamorphic testing module for DNN-based autonomous driving systems, where the metamorphic relations are defined such that regardless of how the driving scenes are synthesized to cope with weather conditions, the driving behaviors are expected to be consistent with those under the corresponding original driving scenes.

Input Validation. Input Validation aims at ensuring that only properly formed data can be accepted by an information system, and preventing malformed data leading systems errors. Input Validation has be applied to enhance the robustness of web application [1, 25]. In this paper, DeepRoad develops a distance-based input validation framework for DNN-based autonomous driving systems, where the key idea is that a valid input image is similar to a part of the images in the training dataset, and the similarity can be measured by the distance in a non-linear low-dimension space. To enhance the systems' security, the images will be rejected if their distance is greater than a given threshold.

Testing and verification of DNN-based autonomous driving systems. Different from traditional testing practices for DNN models [28, 39], a recent set of approaches (such as DeepXplore [31] and DeepTest [38]) utilize differential and metamorphic testing algorithms for identifying inputs that trigger inconsistencies among different DNN models, or among the original and transformed driving scenes. Although such approaches have successfully found various autonomous driving system issues, there still lack approaches that can test DNN-based autonomous driving system with diverse and realistic synthesized driving scenes. Moreover, DeepSafe [19] focuses on automatically identifying safe regions of the input space, within which the network is robust against adversarial perturbations.

GAN-based Image Translation. GAN-based domain adaption has been recently shown to be effective in unsupervised image-to-image translation [21, 27, 43, 45]. CycleGan [45], DiscoGAN [21] and DualGan [43] propose the similar idea that image-to-image translation should satisfy the cycle consistency, where an image from Domain A should be identical when it is translated to Domain B and translated back to A. The experiments show that this extra constraint can make the translated images more realistic. UNIT [27] further assumes that the representations of two domains may be projected to the same vector space (shared latent space), and is constructed based on VAEs and GANs. Specifically, they also apply cycle consistency to the GAN model to regularize the translation.

Moreover, GAN-based domain adaption is also applied for virtual-to-real and real-to-virtual driving scene adaption [26, 42]. DU-drive [42] proposes an unsupervised real to virtual domain unification framework for end-to-end driving. Their key insight is the raw image may contain nuisance details which are not related to the prediction of steering angles, and a corresponding virtual scene can ignore these details and also address the domain shift problem. Grad-GAN [26] is designed to automatically transfer the scene annotation in virtual-world to facilitate real-world visual tasks. In that work, a semantic-aware discriminator is proposed for validating the fidelity of rendered image w.r.t each semantic region.

7 CONCLUSION

In this paper, we propose DeepRoad, an unsupervised learning framework to synthesize realistic driving scenes to test inconsistent behaviors of DNN-based autonomous driving systems, and validate online input images to improve the system robustness. The experimental results on three real-world Udacity autonomous driving models indicate that DeepRoad can successfully detect thousands of inconsistent behaviors. Furthermore, our results also show that DeepRoad can effectively validate input images to potentially enhance the system robustness.

8 ACKNOWLEDGEMENT

This work was supported by the Ministry of Science and Technology of China (Grant No. 2017YFC0804002), Shenzhen Peacock Plan (Grant No. KQTD201611 2514355531), and Science and Technology Innovation Committee Foundation of Shenzhen (Grant No. ZDSYS201703031748284 and No. JCYJ20170817110848086). It was also supported by NSF grants CNS 1527727, CCF-1566589, CNS CAREER 1750263, and CCF-1704790. The authors thank Shiwei Yan for the support of evaluations, and thank Chenguang Liu, Meng Li, Yibo Lin and anonymous reviewers for the valuable comments.

REFERENCES

[1] 2013. Open Web Application Security Project: Data Validation. https://www.owasp.org/index.php/Data_Validation. Accessed: Jun. 2018.

[2] 2014. Tesla Autopilot System. https://www.tesla.com/autopilot. Accessed: Jun. 2018.

[3] 2016. Final leaderboard of Udacity Challenge 2. https://github.com/udacity/self-driving-car/tree/master/challenges/challenge-2. Accessed: Jun. 2018.

[4] 2016. Steering angle model: Autumn. https://github.com/udacity/self-driving-car/tree/master/steering-models/evaluation. Accessed: Jun. 2018.

[5] 2016. Steering angle model: Chauffeur. https://github.com/udacity/self-driving-car/tree/master/steering-models/community-models/chauffeur. Accessed: Jun. 2018.

[6] 2016. Steering angle model: Rwightman. https://github.com/udacity/self-driving-car/tree/master/steering-models/evaluation. Accessed: Jun. 2018.

[7] 2016. Udacity pre-trained Models. https://github.com/udacity/self-driving-car/tree/master/steering-models/evaluation. Accessed: Jun. 2018.

[8] 2016. Udacity self driving car. https://github.com/udacity/self-driving-car. Accessed: Jun. 2018.

[9] 2018. Tesla Model S crash. https://www.wired.com/story/tesla-autopilot-why-crash-radar. Accessed: Jun. 2018.

[10] 2018. Uber's Self-Driving Cars Were Struggling Before Arizona Crash. https://www.nytimes.com/2018/03/23/technology/uber-self-driving-cars-arizona.html. Accessed: Jun. 2018.

[11] Paul Ammann and Jeff Offutt. 2016. *Introduction to software testing*. Cambridge University Press.

[12] Shane Barratt and Rishi Sharma. 2018. A Note on the Inception Score. *arXiv preprint arXiv:1801.01973* (2018).

[13] Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer. http://research.microsoft.com/en-us/um/people/cmbishop/prml/

[14] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. 2016. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316* (2016).

[15] Tsong Y Chen, Shing C Cheung, and Siu Ming Yiu. 1998. *Metamorphic testing: a new approach for generating next test cases*. Technical Report. Technical Report HKUST-CS98-01, Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong.

[16] Leon Gatys, Alexander S Ecker, and Matthias Bethge. 2015. Texture synthesis using convolutional neural networks. In *Advances in Neural Information Processing Systems*. 262–270.

[17] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. 2016. Image style transfer using convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*. IEEE, 2414–2423.

[18] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

[19] Divya Gopinath, Guy Katz, Corina S Pasareanu, and Clark Barrett. 2017. Deepsafe: A data-driven approach for checking adversarial robustness in neural networks. *arXiv preprint arXiv:1710.00486* (2017).

[20] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. 2016. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*. Springer, 694–711.

[21] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jungkwon Lee, and Jiwon Kim. 2017. Learning to discover cross-domain relations with generative adversarial networks. *arXiv preprint arXiv:1703.05192* (2017).

[22] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[24] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436.

[25] Nuo Li, Tao Xie, Maozhong Jin, and Chao Liu. 2010. Perturbation-based user-input-validation testing of web applications. *Journal of Systems and Software* 83, 11 (2010), 2263–2274.

[26] Peilun Li, Xiaodan Liang, Daoyuan Jia, and Eric P Xing. 2018. Semantic-aware Grad-GAN for Virtual-to-Real Urban Scene Adaption. *arXiv preprint arXiv:1801.01726* (2018).

[27] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. 2017. Unsupervised image-to-image translation networks. In *Advances in Neural Information Processing Systems*. 700–708.

[28] Alexis C Madrigal. 2017. Inside waymo's secret world for training self-driving cars. *The Atlantic* (2017).

[29] William M McKeeman. 1998. Differential testing for software. *Digital Technical Journal* 10, 1 (1998), 100–107.

[30] Christian Murphy, Gail E Kaiser, Lifeng Hu, and Leon Wu. 2008. Properties of Machine Learning Applications for Use in Metamorphic Testing. In *SEKE*, Vol. 8. 867–872.

[31] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. 2017. Deepxplore: Automated whitebox testing of deep learning systems. In *Proceedings of the 26th Symposium on Operating Systems Principles*. ACM, 1–18.

[32] Dean A. Pomerleau. 1989. Advances in Neural Information Processing Systems 1. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, Chapter ALVINN: An Autonomous Land Vehicle in a Neural Network, 305–313.

[33] Haşim Sak, Andrew Senior, and Françoise Beaufays. 2014. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Fifteenth annual conference of the international speech communication association*.

[34] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*. 2234–2242.

[35] S. Segura, G. Fraser, A. B. Sanchez, and A. Ruiz-Cortes. 2016. A Survey on Metamorphic Testing. *IEEE Transactions on Software Engineering* 42, 9 (Sept 2016), 805–824.

[36] Sergio Segura, Gordon Fraser, Ana B Sanchez, and Antonio Ruiz-Cortés. 2016. A survey on metamorphic testing. *IEEE Transactions on software engineering* 42, 9 (2016), 805–824.

[37] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[38] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. 2018. DeepTest: Automated Testing of Deep-Neural-Network-driven Autonomous Cars. In *Proceedings of the 40th International Conference on Software Engineering, Gothenburg, Sweden, May 27 - June 3, 2018 (ICSE 2018)*.

[39] Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. 2016. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

[40] Xiaoyuan Xie, Joshua Ho, Christian Murphy, Gail Kaiser, Baowen Xu, and Tsong Yueh Chen. 2009. Application of metamorphic testing to supervised classifiers. In *Quality Software, 2009. QSIC'09. 9th International Conference on*. IEEE, 135–144.

[41] Xiaoyuan Xie, Joshua WK Ho, Christian Murphy, Gail Kaiser, Baowen Xu, and Tsong Yueh Chen. 2011. Testing and validating machine learning classifiers by metamorphic testing. *Journal of Systems and Software* 84, 4 (2011), 544–558.

[42] Luona Yang, Xiaodan Liang, and Eric Xing. 2018. Unsupervised Real-to-Virtual Domain Unification for End-to-End Highway Driving. *arXiv preprint arXiv:1801.03458* (2018).

[43] Zili Yi, Hao Zhang, Ping Tan, and Minglun Gong. 2017. Dualgan: Unsupervised dual learning for image-to-image translation. *arXiv preprint* (2017).

[44] Zhi Quan Zhou, DH Huang, TH Tse, Zongyuan Yang, Haitao Huang, and TY Chen. 2004. Metamorphic testing and its applications. In *Proceedings of the 8th International Symposium on Future Software Technology (ISFST 2004)*. 346–351.

[45] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593* (2017).